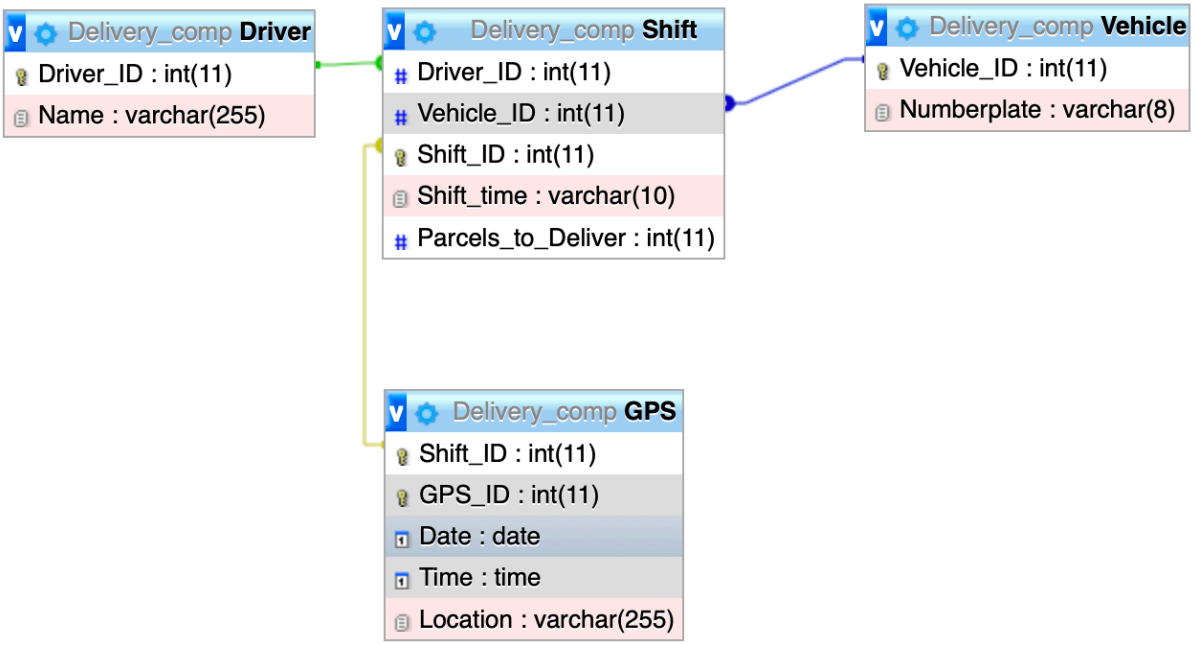
Jaden Rogers – Scalable Database Systems Assignment 1 Report

Design Process

To begin with I extracted all the relevant information from the brief. This resulted in the entities Driver, Vehicle and Parcel. Then I worked out what attributes I will need to meet the query specifications. These included driver names, the time and date of the shift, parcels to be delivered in each shift and then GPS locations for each hour including the time for each shift. After I finished extracting the information out of the brief, I started working out the relationships between the entities. As it states that each driver will be assigned a car each morning, I thought that it will be many-to-many relationship between the driver and vehicle entities because each vehicle will be used by different drivers throughout the week and each driver will use different cars throughout the week. This is how I created the Shift entity as each driver will have a shift each day and each shift will use a vehicle. This entity will contain the attributes describing the shift like Shift time and the parcels to deliver. Each shift will also have 4 GPS updates for the location of the vehicle so it will be a one-to-many relationship. The GPS entity will have a composite key made of the Shift id and the GPS id so that each entry will be uniquely identified but you will still be able to find all the GPS updates for a certain shift. My database design is normalised to at least 2NF as all attributes are fully dependant on the primary key. In Driver, Name is solely describing the Driver, just as Numberplate is solely describing Vehicle in that table. Moreover, in shift, all attributes are fully dependant on the Shift_ID as every shift will have its own ID and combination of vehicle, driver, Shift_time and Parcels_to_Deliver. Finally, in the table GPS, which has a composite key of Shift_ID and GPS_ID, the same rule applies as the attributes refer to a specific combination of the shift_ID and GPS_ID for each hourly update.



When planning the database, I used a more straightforward approach to the shifts to make inputting the data more efficient however, the database is designed to be able to handle the data just as well if there are changes to this. For example, if new drivers and vehicles are employed, delivery routes are changed, and parcel numbers are changed.

	Vehicle	Monday	Tuesday	Wednesday	Thursday	Friday
Morning	**1**	Driver 1	Driver 1	Driver 1	Driver 1	Driver 1
Morning	**2**	Driver 2	Driver 2	Driver 2	Driver 2	Driver 2
Morning	**3**	Driver 3	Driver 3	Driver 3	Driver 3	Driver 3
Morning	**4**	Driver 4	Driver 4	Driver 4	Driver 4	Driver 4
Morning	**5**	Driver 5	Driver 5	Driver 5	Driver 5	Driver 5
Afternoon	**1**	Driver 6	Driver 6	Driver 6	Driver 6	Driver 6
Afternoon	**2**	Driver 7	Driver 7	Driver 7	Driver 7	Driver 7
Afternoon	**3**	Driver 8	Driver 8	Driver 8	Driver 8	Driver 8
Afternoon	**4**	Driver 9	Driver 9	Driver 9	Driver 9	Driver 9
Afternoon	**5**	Driver 10	Driver 10	Driver 10	Driver 10	Driver 10

Queries and Procedures

Query 4.1

From the brief, I gathered that the query needed to take the vehicle number and time of day, then return the name of the driver and their location at the time given. To do this it needs to access the Driver, Shift and GPS entities using inner joins as it needs to return the values which meet all three parameters. As you can see, it returns the specified attributes successfully and I also show the inputted data, so the user is able to make sure they are checking the correct vehicle number, time and date.

```
SELECT Driver.Name, Shift.Vehicle_ID, GPS.Location, GPS.Date, GPS.Time
FROM  Driver
INNER JOIN Shift ON Driver.Driver_ID=Shift.Driver_ID
INNER JOIN GPS ON Shift.Shift_ID=GPS.Shift_ID
WHERE Shift.Vehicle_ID = 3 AND GPS.Time = '11:00:00' AND GPS.Date = '2021-01-21';
```

Name	Vehicle_ID	Location	Date	Time
Steve	3	Brampton	2021-01-21	11:00:00

Query 4.2

This query needed to use drivers name, and the date, to show how many parcels they will be delivering that day. Similarly to the first one, it needs to access the Driver, Shift and GPS entities using inner joins as it needs to return the values which meet both of the parameters. It returns any Parcels_to_Deliver attributes where the Name is “Candice” and the date is 2021-01-23. As Candice has one shift one that day there is one entry on the resultant table.

```
SELECT DISTINCT Driver.Name, Shift.Parcels_to_Deliver, GPS.Date
FROM  Shift
INNER JOIN Driver ON Shift.Driver_ID=Driver.Driver_ID
INNER JOIN GPS ON Shift.Shift_ID=GPS.Shift_ID
WHERE Driver.Name = 'Candice' AND GPS.Date = '2021-01-23';
```

Name	Parcels_to_Deliver	Date
Candice	46	2021-01-23

Query 4.3

The way my database is designed made this a very simple query to complete as it just needs to access all the Name attributes from the Driver entity. This query will display the names of all the drivers in the database in the default order of how they were added originally.

```
SELECT Name FROM Driver;
```

Name
Pete
John
Steve
Perry
Malcolm
Phineas
Ferb
Candice
Junior
Julio

Query 4.4

This query needs to access the Driver and Shift entities because it selects any drivers that have only driven morning hour shifts. At first, I used a simple, singular SELECT statement that returns all drivers that have morning shift. Due to the way my shifts were set up this appeared to meet the query as all the drivers do the same shift time throughout the whole week. However, when I looked back, I noticed that a driver name would still be returned even if they had done an afternoon shift at some point which is not what the brief specified. To fix this, I added and EXCEPT statement which excluded any drivers that have ever done even one afternoon shift from being returned.

```
SELECT DISTINCT Driver.Name
FROM  Driver
INNER JOIN Shift ON Driver.Driver_ID=Shift.Driver_ID
WHERE Shift.Shift_time = 'Morning'
EXCEPT
SELECT DISTINCT Driver.Name
FROM  Driver
INNER JOIN Shift ON Driver.Driver_ID=Shift.Driver_ID
WHERE Shift.Shift_time = 'Afternoon';
```

Name
Pete
John
Steve
Perry
Malcolm

Query 4.2 (Procedure)

This procedure works in a very similar way as the query that I used before except I have included parameters that allow the user to call it with the Name and Date they want instead of having to write a whole new query each time. This result was created by calling the store procedure with parameters (“Candice”, “2021-01-24”).

```
DELIMITER //
  CREATE OR REPLACE PROCEDURE Driver_Parcels (Driver_Name varchar(10),Day DATE)
  BEGIN
    SELECT DISTINCT Driver.Name, Shift.Parcels_to_Deliver, GPS.Date
    FROM  Shift
    INNER JOIN Driver ON Shift.Driver_ID=Driver.Driver_ID
    INNER JOIN GPS ON Shift.Shift_ID=GPS.Shift_ID
    WHERE Driver.Name = Driver_Name AND GPS.Date = Day;
END;
//
```

Name	Parcels_to_Deliver	Date
Candice	62	2021-01-24

Query 4.1 (Procedure)

This Procedure is also very similar to the original query and I have also added parameters to this one. As you can see, I change the delimiter back to “;” as I had already changed it when writing the other procedure as I did that one first. This result was produced by calling the procedure with the parameters (5, “12:00:00”, “2021-01-23”)

```
  CREATE OR REPLACE PROCEDURE Driver_Hourly_Location (Car_Num INT, Hour Time, Day DATE)
  BEGIN
    SELECT Driver.Name, Shift.Vehicle_ID, GPS.Location, GPS.Date, GPS.Time
    FROM  Driver
    INNER JOIN Shift ON Driver.Driver_ID=Shift.Driver_ID
    INNER JOIN GPS ON Shift.Shift_ID=GPS.Shift_ID
    WHERE Shift.Vehicle_ID = Car_Num AND  GPS.Time = Hour AND GPS.Date = Day;
END;
//

DELIMITER ;
```

Name	Vehicle_ID	Location	Date	Time
Julio	5	Ramsey	2021-01-23	12:00:00

Reflection

In my opinion I have produced an efficient solution to the problem, however, it could be improved as it doesn't track individual parcels. I would include this functionality by creating a 'Parcel' table with a composite key consisting of Shift_ID and Parcel_ID, then the attributes: Postcode, street number, street name and Name of reciever. It would be a one-to-many relationship with Shift and it would work in a similar way to the GPS table.

Another way my database could be improved is to add security. To do this, if there are multiple users of the database, I could assign different clearance levels to each user. This would improve the security as you wouldn't want everyone who accesses the database to be able to alter information in it or the structure of the relationships and attributes of the different tables. This could harm the database as an inexperience user could break the whole thing by accident. This ability would be limited to as few as possible users. For normal users, I could grant them limited access so they can only pull data from it as that is the only thing they need it for. Other users, like admin employees, could be granted access to only add data into the tables as they would need to add in new employees or vehicles' details and data about the shifts and parcels.